Eye-Spy
Developer Manual

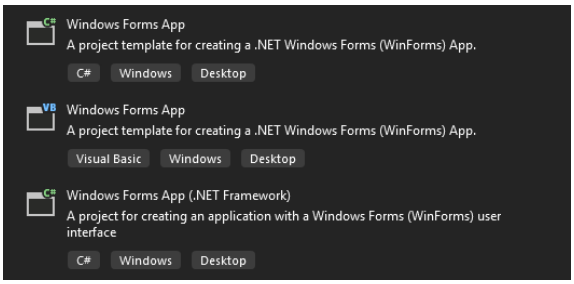
Jack Strauss

CSCI460

2026

Setting up Tobii Pro Spark Project

For this section, I will reference the setup guide written by SNC alum Taylor Wesolowski. It describes how to set up a project in Visual Studios 2022 for the Tobii Pro Spark. It was what I used when setting up my application.

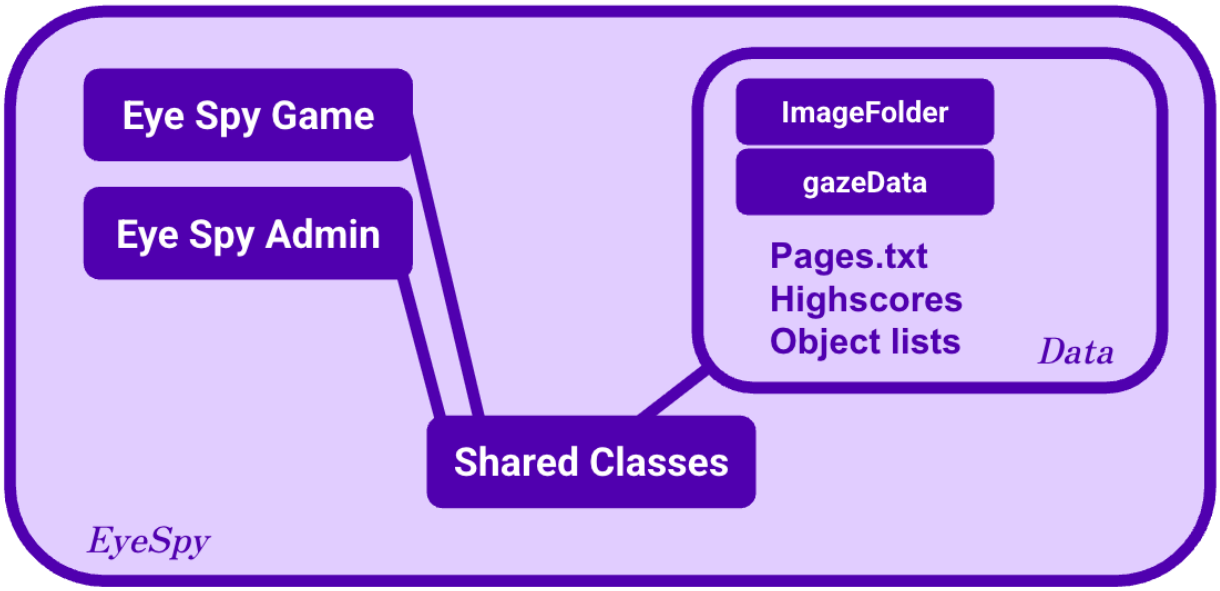


I will also just make it clear, because this tripped me up, *make sure you choose the one that says (.NET framework) in the title. The one without it, that just says Windows Forms App will not be compatible.*

Taylor Wesolowski
(SNC Computer Science / Mathematics Graduate - Class of 2024): https://compsci04.snc.edu/cs460/2024/taylorwesolowski/readMe.pdf

Data Structure

All of the files for Eye Spy live in a parent folder. Inside of this folder, there are four folders, the Eye Spy Game, the Eye Spy Admin Application, the Shared Classes Repository, and the data files (Images, Object lists, etc.)



In the Data folder, there is the ImageFolder, which holds the key icons(in the objects folder) as well as the base images for each Page. The gazeData Folder, which contains the recorded data for each run of the game to be replayed.

The Data folder also contains the pages.txt, which contains a list of pages, as well as the highscores list and object list for each page.
To reference any of these directories, make sure that you have SharedClasses included in your project. Then you can reference them by using AppPaths.(Directory). For example, to write to the ImageFolder, you would reference it as AppPaths.ImageFolder.

The full list of folders in AppPaths is:

- Base Folder (Data folder)
- ImageFolder
- ObjectFolder
- DataFolder (gazeData)
- Pages

Classes

Page

Used to display images on the Main Menus

String display: Name displayed in menus
String Name:name used in files
Image BW: Black and White image
Image Color: Color Image
Bool isColor: Is the page in color or not?

ImageHelper

Contains a method to load images safely, makes sure we don't attempt to copy an image to the destination that it already exists in.

HighScore

These inhabit the high score box in the postGame form

String name
Double time
String filename: references filename for gazeData associated with the score
String display: text displayed on the score list

ImageBackground

Image displayed during gameplay and postgame. Has two modes, Color and Black and White. This changes how finding an object is visualized. Also changes if Black and White images of the the Base Image and Key Icons are necessary.

String Name
Image BlackWhiteImage
Image ColorImage
Image BaseImage: whatever image is displayed to screen, changes based on if color or not
List of HiddenObjects objects
String filepath: path of objects list
Bool isColor

HiddenObject

These live in an ImageBackground, are cross referenced by data from eye tracker.

Rectangle rect: Boundary of object
String name
Bool found: initialized to false
Int lookcounter: how long has object been looked at
Image KeyIcon
Image ColorIcon
String pageName: what page is it associated with

GazeSample

Used in constructing and reconstructing replay data for a run of the game, represents a single sample of gaze data received by the eye tracker.

Long timestamp: time in milliseconds
Float X
Float Y

Main Page

In the main page, the dropdown list reads the list of pages, and creates pages based on each line. The images are found by grabbing the image at the directory of AppPaths.ImageFolder.(pageName).ext/(pageName)Color.ext. The display name of each page is shown on the dropdown. Whenever the selected item changes on the dropdown, the page's image is displayed in the picture box to the right, but it is first pixelated.

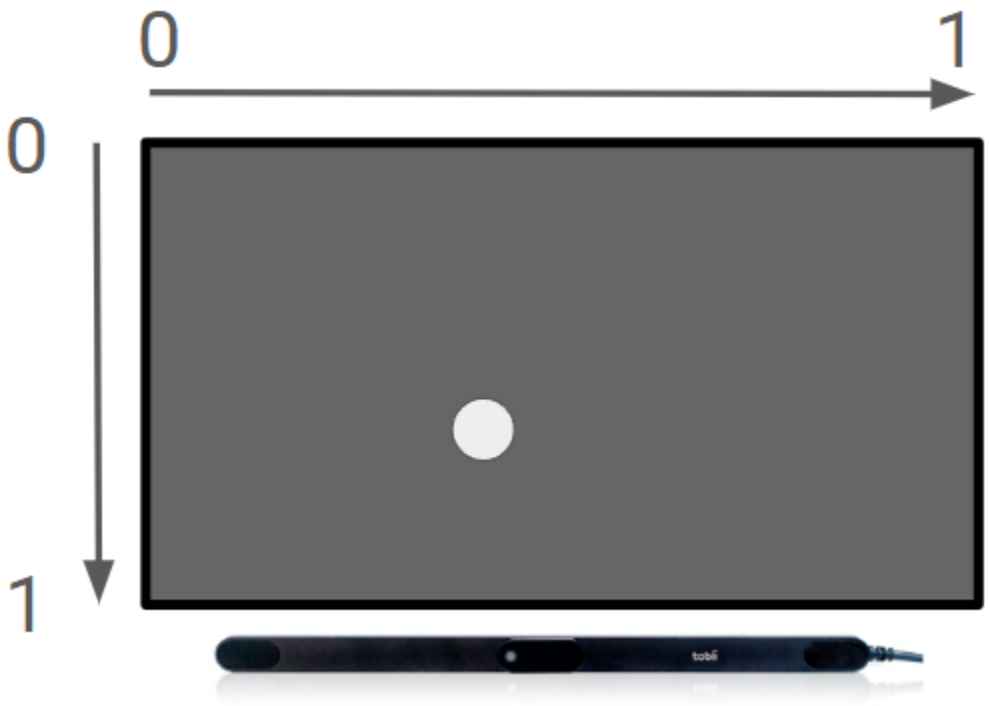
When the play button is clicked, it opens a gameForm with the name of the currently selected page.

The High Scores button opens a postGame form with the default information (first page in pages.txt, top score)

Game Form

The gameForm receives a name of the page and a bool of if the page is in color from the main form, it creates an ImageBackground with the name. Then we attempt to load the eyetracker, if it does not load, we close the form. Similarly, we then ask for a name, and if it is empty, we exit. Lastly, we then start the timer.

Receiving Gaze Data



In Eye Spy, Gaze Data is requested from the Tobii Pro Spark on every timer tick. Which happens every 20 milliseconds (~50 sec). We receive an X and and Y, these are both normalized to the screen. This means that they come in as floats. This assumes that the width and height of your display run from 0 to 1. For example, in the above diagram, the white dot would be at:

X: 0.4193339 Y: 0.5286691

Every third set of valid coordinates is saved to a list which will be written to a file for replay once all objects are found.

Drawing to the screen/Bitmaps

To initialize the game, we draw a static layer first. We have two bitmaps, one for the base layer, and one for the objects which will be drawn on top. These are then drawn to the screen during every call of onPaint. This is much more efficient than drawing every object individually every frame, as we only make two draw calls instead of one for every object. Whenever an object is found:

- Color
    - Rectangle at object's coordinates is drawn on object bitmap
    - Gray rectangle is drawn on top of object's key icon on object bitmap
- Black and White
    - Object's boundary is used to grab a rectangle from the color image, and it is drawn to the object bitmap
    - Object's color icon is drawn on top of old icon on the object bitmap
    -

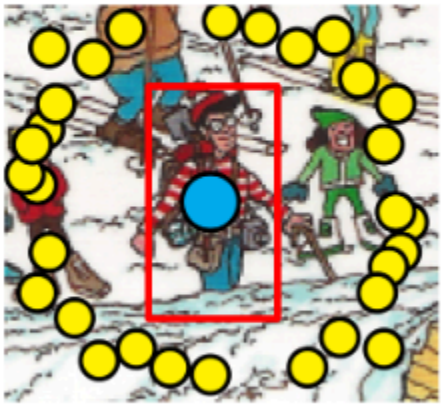
The Main Image is drawn centered within a rectangle which has a 10% margin from the left of the screen, has a width of 50% of the screen, and is fit to the width of the screen. The image does not change aspect ratio. Whenever the coordinates matter, the images are drawn in the exact same way. (postGame, addNewPage)

During the onPaint method, we also draw a red dot to the screen at the coordinates of the gaze data we received to indicate to the user where they are looking.

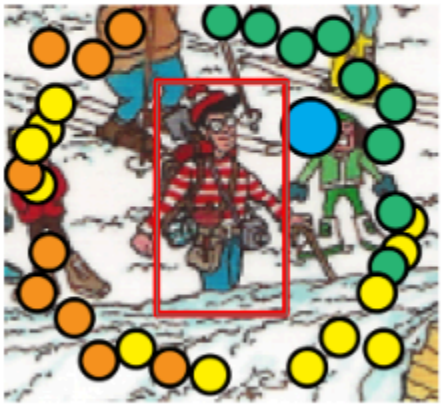
It is also possible that we may be in a paused state, in this case, a gray rectangle is drawn instead of the base layer, the object bitmap is still drawn on top.

Finding Objects

We keep a queue of the last 30 received gazeSamples. On every tick, we do a weighted average, in which the most recent 10 samples are multiplied to make up 60% of the average, the next 10 make up 30%, and the least recent make up 10%. This gives the most recent points more gravity.



Basic Average



Weighted Average

We then check this average against the boundaries of every object, and if it is in the boundary, the lookcounter of the object is increased. If the average stays in the boundary for 15 ticks, the object is considered found. If the average ever leaves the boundary, the lookcounter is set to zero.

postGame

Once all objects are found, a postGame form is opened. It is passed the Username, the final TimerTime, the name of the page that was played, the filepath for the replay data, as well as if the page is in color or not.

If we enter from the main menu, none of that information is necessary, and the scores will be loaded without the new data.

We then draw the current image to the screen and load the scores into the highscore box. In LoadScores, we create a list of HighScores from the page's highscore list document and add this list to the box.

Whenever the selected item in the highscorebox is changed, we replay the gaze data in that score's gazeData file and is loaded into a list of GazeSamples. We then replay the data, by starting a timer, and comparing the current time of the timer to the milliseconds of the gazeSamples. If that sample"s millisecond count is smaller than the timer, a red dot is drawn to the screen at its coordinates. Loading the highscore box automatically selects an item, so a replay will automatically start upon loading the form. If we enter with a user's information, their final time will be displayed, and if their time was in the top ten, their score is automatically selected.

There is also a dropdown menu to switch between the different pages, which will reload the scores with the new page's score list and draw the new image to the screen.

adminForm

The dropdown works in the exact same way as the main page of the game.

There is an edit button, which opens the addNewPage form, with the information of the selected page. A delete button, which deletes the page from pages.txt, deletes the images associated with the page, as well as the page's highscore and object lists.

The Add New Page button opens a blank addNewPage form

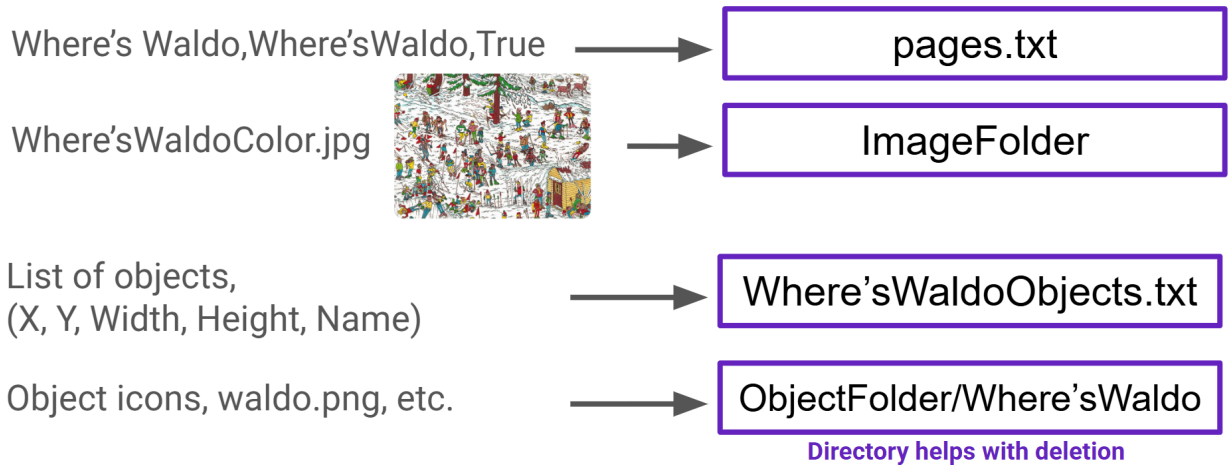
addNewPage

The addNewPage has the information organized on the right side.

- Text box for page name
- Checkbox for if it is a color image or not
- Buttons to upload the necessary images
- Previews for the images that have been uploaded
- List of current objects

On the right is a picture box which will hold the color image that has been uploaded. This picturebox is set to the same dimensions that the image is drawn in the game. It uses the zoom picture mode so that the image fills the space without changing aspect ratio.

Once the user enters all necessary information for an object, a new HiddenObject is added to the list of objects. The user can delete an object, which will remove the object from the list. When the finish button is pressed, the following information is written/copied to the following destinations(Using a color page named Where's Waldo as an example). All invalid characters as well as commas are removed from the name for file naming purposes.



Color images are copied as (pageName)Color.ext, and black and white images are copied as (pageName).ext. These are both copied to the ImageFolder.